G E M I N I   M I C R O C O M P U T E R S

GM860

`BYTEWIDE´

EPROM

PROGRAMMER

MANUAL

ISSUE 1

14/08/84

The following section gives more details on the EPROM types supported by the GM860 software. Anyone wishing to program an EPROM should check the programming requirements of the device as given in the manufacturer's data sheet against those used by the GM860 software to ensure that they match.

In general it is just a matter of selecting the option that uses the correct programming voltage for the device, (eg 27128 for a programming voltage of 21v, or 27128A for a programming voltage of 12.5v). However in the case of the 2k x 8 EPROMs, (before uniform standards appeared), there were at least two types of device that required different programming stratagies. Devices compatible with the Intel 2716 are supported by the GM860 software and hardware, (eg the Texas TMS2516), but the Texas TMS2716 is not.

QUICK SUMMARY

Menu Option	Prog. volts and pin.	Programming technique
2716	25v @ 21	50ms pulse @ 18
2732	25v @ 20	50ms pulse @ 18
2732A	21v @ 20	50ms pulse @ 18
2764	21v @ 1	IPA @ 27
2764A	12.5v @ 1	IPA @ 27
27128	21v @ 1	IPA @ 27
27128A	12.5v @ 1	IPA @ 27
27256	12.5v @ 1	IPA @ 20
27256T	21v @ 1	IPA @ 20

IPA - Intelligent Programming Algorithm.
(Fast programming technique).

EPROM PINOUTS

27256	27128	2764	2732	2716			2716	2732	2764	27128	27256
Vpp	Vpp	Vpp			[1	28]			Vcc	Vcc	Vcc
A12	A12	A12			[2	27]			Pgm	Pgm	A14
A7	A7	A7	A7	A7	[3	26]	Vcc	Vcc	nc	A13	A13
A6	A6	A6	A6	A6	[4	25]	A8	A8	A8	A8	A8
A5	A5	A5	A5	A5	[5	24]	A9	A9	A9	A9	A9
A4	A4	A4	A4	A4	[6	23]	Vpp	A11	A11	A11	A11
A3	A3	A3	A3	A3	[7	22]	/OE	/OE	/OE	/OE	/OE
A2	A2	A2	A2	A2	[8	21]	A10	A10	A10	A10	A10
A1	A1	A1	A1	A1	[9	20]	/CE	/CE	/CE	/CE	/CE
A0	A0	A0	A0	A0	[10	19]	O7	O7	O7	O7	O7
O0	O0	O0	O0	O0	[11	18]	O6	O6	O6	O6	O6
O1	O1	O1	O1	O1	[12	17]	O5	O5	O5	O5	O5
O2	O2	O2	O2	O2	[13	16]	O4	O4	O4	O4	O4
GND	GND	GND	GND	GND	[14	15]	O3	O3	O3	O3	O3

T A B L E   O F   C O N T E N T S

1. Introduction

The Gemini GM860 EPROM programmer is primarily designed for use with Gemini computer systems such as the Galaxy, but may also be used with Quantum, Kenilworth, Gemini MultiBoard and Nascom systems and other 4MHz Z80A based systems equipped with a Z80 PIO device. The GM860 may program 2716, 2732, 2732A, 2764, 27128, and 27256 type EPROMs, utilising a high-speed programming algorithm for 2764 and larger devices.

2. The Hardware

The GM860 EPROM programmer is self-contained with its own power supply, requiring only connection to the mains and a single cable connection to the host computer. The front of the unit is fitted with a mains ON/OFF switch and a fuse holder. This is fitted with a 20mm 315mA fuse and should it ever need replacing the same type should be used. The top of the unit is fitted with a 28-pin Zero Insertion Force (ZIF) socket and two indicators - one indicating when the unit is switched on, the other indicating when any access, either read or write, is being made to the EPROM socket. The rear of the unit is fitted with a mains supply cable and computer connection socket.

2.1. Mains Connection

The GM860 is intended for 220/240V mains operation and is supplied with a suitable mains cable. The mains cable requires the attachment of a mains plug. The cable is to standard European specification - i.e. green/yellow is Earth, blue is Neutral, brown is Live. A low current fuse should be installed in the mains plug, say 3 Amp.

2.2. Computer Connection

Connection to the host computer will depend on the type of computer being used.

2.2.1. Gemini and Quantum Computer Systems

When used with complete Gemini or Quantum Computer Systems, such as the Gemini Galaxy or Quantum 2000 range, connection will normally be made using the cable supplied. The flat ID connector on the cable should be plugged into the corresponding connector on the rear of the GM860 unit, ensuring that it is connected with the correct orientation. Pin 1 of the cable is indicated by a coloured stripe and by a triangular indentation on the connector. Pin 1 on the GM860 unit is indicated by the white trianglular marking. The other end of the cable is fitted with a Centronics type 36-way connector, and this should be plugged into the 'Parallel Printer' socket on the rear of the computer. Only one orientation is possible.

NOTE - if the computer is equipped with a MultiNet interface board then it must also be equipped with an I/O board (GM816, GM848 or Nascom I/O) in order to provide the Centronics interface.

2.2.2. Gemini MultiBoard and Nascom Systems

When used with Gemini MultiBoard or Nascom systems connection of the GM860 unit is made to a Z80 PIO device in the system. The Gemini GM669 cable may be used for this, this cable consisting of a 26-way IDS connector (Gemini/Nascom PIO connector compatible) connected to a 36-way Centronics type connector.

The 26-way connector may be connected directly to a PIO port on the Gemini GM811 CPU board, GM813 CPU/RAM board, GM816 Multi I/O board, GM848 Serial/Parallel I/O board, Nascom 2, or Nascom I/O board. The Centronics type connector is connected via the cable supplied with the GM860 unit, as described in the section above.

For those wishing to make up their own cables the necessary connections are as follows:

26-way Gemini/Nascom PIO socket	34-way connector on GM860
1	13
2	11
3	15
4	9
5	17
6	7
7	N.C.
8	5
9	*
10	3
11	*
12	N.C.
13	21
14	*
15	1
16	*
17	N.C.
18	*
19-25	N.C.
26	*

* - ALL even numbered pins of the 34-way connector, plus pins 19, 23, 25, 27, 29, 31, 33 are 0 volt connections, and so any of these may be used for these connections.

N.C. - No connection

2.2.3. Other Computers

Connection may be made to other 4MHz Z80A based computers that are fitted with a Z80 PIO device, provided that the four Z80 I/O ports used by the PIO are contiguous, and provided that they are addressed in the sequence:

Port A data
Port B data
Port A control
Port B control

Connection details are provided below:

PIO connection	34-way connector on GM860
A0	21
A1	15
B0	10
B1	8
B2	6
B3	4
B4	2
B5	1
B6	3
B7	5
/ASTB	*
/BSTB	*
0 volts	*

* - ALL even numbered pins of the 34-way connector, plus pins 19, 23, 25, 27, 29, 31, 33 are 0 volt connections, and so any of these may be used for these connections.

2.3. Inserting and Removing EPROMs

The GM860 programmer should be switched on and the PROG program run to initialise the programmer before inserting EPROMs into the ZIF socket. Note that EPROMs may only be inserted and removed when the ´Busy´ indicator is OFF.

To insert an EPROM, raise the arm of the ZIF socket to open the contacts, place the EPROM into the socket, and then lower the arm of the ZIF. To remove the EPROM raise the arm of the ZIF socket again, and lift the EPROM from the socket.

28-pin EPROMs - 2764, 27128, and 27256 devices are all 28-pin components and should be inserted into the ZIF socket with pin 1 of the EPROM corresponding to the ´28-Pin´ legend on the GM860 unit.

24-pin EPROMs - 2716 and 2732 devices are 24-pin components and should be inserted into the ZIF socket with the pin 12/13 end of the EPROM as far towards the front of the GM860 as it can be inserted i.e. with pin 1 of the EPROM going into pin 3 of the ZIF socket as indicated by the ´24-Pin´ legend.

```
28 Pin ---> +----+                              1 +----+ 28
            |    |                                |    |
24 Pin ---> |    |           1 +----+ 24          |    |
            |ZIF |             | 24 |             | 28 |
            |    |             |    |             |    |
            |    |             | P  |             | P  |
            |    |             | I  |             | I  |
            |    |             | N  |             | N  |
            +----+ - - - - 12 +----+ 13 - - - 14 +----+ 15
```

3. Software supplied

Three CP/M programs are provided in conjunction with the Gemini GM860 EPROM programmer. (See the Appendix for disk format details.) These are:

PROG.COM	- The program to use to program EPROMs in GM860
CPROG.COM	- A configuration program for PROG.COM
PROGINIT.MAC	- The assembly language interface routines for GM860 for those who wish to write their own driver.

NOTE - the programming pulse timings provided by the above software are based on the host system running at 4MHz. If the host system is running at any other speed then the GM860 will not operate correctly.

3.1. CPROG.COM

This configuration program allows you to change two options within PROG.COM. It is menu driven, and should be simple to operate.

3.1.1. Default EPROM type

The first item you may change is the EPROM type that PROG defaults to when it is first run. PROG does allow you to change EPROM types while it is running, but this feature is provided in CPROG for the convenience of those who normally use EPROMs other than the current default value in PROG.

3.1.2. PIO base address

The second option allows you to change the address of the PIO used by PROG to drive the EPROM programmer. If you have the programmer connected to the standard Centronics port of a Gemini or Quantum computer system ,such as the Gemini Galaxy, you will not need to use this option, unless the system is fitted with a MultiNet interface (see section 2.2.1). If you have connected the Programmer to a PIO other than the one on the main CPU board, (e.g. to a PIO on an I/O board), then you should select this option and enter the base address (in Hex) of the PIO used.

NOTE - Each PIO in an 80-BUS system normally occupies four consecutive addresses in the I/O address space. What should be entered here is the lowest of the four addresses occupied by the PIO. The software assumes that the PIO is connected in the conventional Gemini/Nascom manner where the ports go:- A-side data, B-side data, A-side control, B-side control.

3.2. PROG.COM

PROG is the program that you use to program EPROMs in conjunction with GM860. The program is invoked by typing **PROG.** No additional options or parameters are required. PROG responds by putting up a heading and a main menu.

While PROG is running you will find that the ESCape key acts as an abort key and will return you to the main menu.

3.2.1. The Heading

Across the top line you will see a heading which includes the version number of the software. Below this there is a status line. This shows three things:
1) Whether there is any data in the memory buffer maintained by PROG.
2) The type of EPROM selected for Read/Write operations.
3) The name of the last file Read/Written to/from the memory buffer.

3.2.2. The main menu

This is shown below the heading and provides you with ten options, the appropriate option being selected by entering a number in the range 0-9. The various options are covered below.

3.2.2.1. 0 - Exit the program

This returns you immediately to CP/M. Any data currently held in the buffer is lost.

3.2.2.2. 1 - Change EPROM type

PROG puts up a sub-menu from which you can choose the appropriate EPROM type to match the one you wish to program. Note it is important to pick the correct type as there are subtle distinctions between the various types. e.g. The 2732A, although interchangable with the 2732 as far as reading is concerned, requires a programming voltage of 21V rather than the 25V of a 2732. Programming a 2732A as a 2732 is a potentially destructive operation.

The type numbers given in the menu are INTEL part numbers. EPROMs from other manufacturers may differ slightly in nomenclature, but it is essential to confirm that the device to be programmed is equivalent to the INTEL part. (e.g. A TMS2716 from TEXAS INSTRUMENTS cannot be programmed by GM860. The TEXAS equivalent to an INTEL 2716 is actually coded as a TMS2516.)

3.2.2.3. 2 - Read EPROM into buffer

The contents of a previously programmed EPROM may be read into the memory buffer by selecting this option. PROG prompts you to insert the EPROM and to press <RETURN>. The entire contents of the EPROM will be then read into the memory buffer. Once the read is complete PROG requests that you remove the EPROM before continuing.

3.2.2.4. 3 - Read FILE into buffer

A disk file may be loaded into the buffer by invoking this option. PROG will first prompt for a file name, and once this has been entered PROG will open the file and read it into the buffer. The display heading will change to include the name of the file that was loaded when you return to the main menu. Error messages will be given if the file cannot be found, or any read error occurs. If the file is greater than the current EPROM size, (e.g. a file of 4.5K is loaded while an EPROM type of 2732 is selected), then a warning message is issued. (The entire file is read in, it is not truncated.)

3.2.2.5. 4 - Edit buffer contents

The contents of the memory buffer may be edited by selecting this option. PROG initially prompts for a start address. Any hexadecimal address within the range of the currently selected EPROM may be entered here. If no address is entered, a default value of 0 is assumed. Once this has been entered PROG displays the address followed by the byte currently stored at that address. As response to the '>' prompt the following commands are accepted:

<RETURN>	- A RETURN on its own steps to the next address in the buffer without altering the byte at the current address.
-	- A 'minus sign' steps back one address in the buffer without altering the byte at the current address.
HH	- If a hexadecimal number (HH) is entered it is stored in the buffer at the current address, and the buffer address is incremented.

"	- A quote mark signifies that the following characters are to be taken literally, and stored at successive locations in the buffer. (e.g. "ABCD<RETURN> would store the hex numbers 41 42 43 44 at the next four locations in the buffer.
/HHHH	- Changes the address to be patched from the current address to HHHH.

Note multiple commands or bytes may be entered on a single line separated by commas or spaces. e.g.

```
                    0890 21>1 2 3 /900 "fred<RETURN>
```

would result in 0890 being set to 01, 0891 to 02, 0892 to 03, and 0900-0903 to the string "fred". N.B. No other commands are recognised after a quote mark - everything that follows it up to the <RETURN> is stored into the buffer.

3.2.2.6. 5 - Display the buffer

The contents of the memory buffer may be displayed by selecting this option. A starting address is prompted for, and once this has been entered the next 256 bytes of the buffer are displayed in the form:

```
XXXX:  HH HH HH HH HH HH HH HH  - HH HH HH HH HH HH HH HH   AAAAAAAAAAAAAAAA
```

where

XXXX	is the address of the first byte displayed. (The remainder occupy consecutive higher addresses.)
HH	are the hexadecimal bytes held in the buffer, and
A..A	are the ASCII character equivalents of the 16 bytes (HH...HH) in the displayed line. Control characters (which are non-printing) are replaced by a period ".".

3.2.2.7. 6 - Write the buffer to a FILE

When this option is selected you will be prompted for the name of the file in which you want to save the buffer contents. In the event of the file already existing you will be given the option of overwriting it with the new data, or you can abort the operation by pressing the <ESC> key.

3.2.2.8. 7 - Write the buffer to an EPROM

This option programs an EPROM with the current contents of the buffer. It programs the entire EPROM, there is no provision for restricting the range of addresses programmed. For EPROMs of type 2764 and larger PROG uses the ´intelligent programming algorithm´ to significantly reduce the programming time for these EPROMs.

3.2.2.9. 8 - Verify EPROM is erased

This option will read the entire EPROM. It will list all locations within the EPROM that contain data other than 0FFh (the erased value). If you find that you have a previously programmed EPROM in the socket, the ^S (Control/S) key may be used to pause the listing, or the <ESC> key to abort the option.

3.2.2.10. 9 - Verify EPROM against buffer

This option will read the entire EPROM, checking it byte by byte against the data held in the memory buffer. It will list all locations where the two are found to differ. The display format is:

```
    AAAA>>>> EE (BB)
```

where AAAA is the address where the error exists, EE is the byte read back from the EPROM, and BB is the byte held in the buffer. If you find that there are a very large number of differences, the ^S (Control/S) key may be used to pause the listing, or the <ESC> key to abort the option.

3.3. Creating ROMable Images

PROG assumes that any file it reads contains a binary image of the EPROM to be programmed. There are various ways that this can be achieved, and the example below utilises Microsoft's M80 macro assembler and L80 linking loader. The example does not use the full power of L80, as it assumes that the object file will be created from a single assembly, not several assemblies that are subsequently linked together by L80.

Let us assume that we wish to create an EPROM for a target Z80 system. The EPROM is to be located at 0F000H and will contain a simple system monitor. The program is coded as usual, but must be preceeded by the command ".phase" (see below). The ".phase" command is an instruction to M80 to assemble the following instructions as absolute code which starts at the address following the ".phase". However this code is to be stored in-line with the current relative address within M80, not at the absolute address.

```
     .Z80                 ; Expect Z80 opcodes
     .phase 0F000h        ; Assemble for address 0F000H
;
;    Sample program
;

       jp   pwron           ; Power-on Jump
pwron: ld   sp,stack        ; Set up the stack
       .....
       .....                ; Main program follows
       end
```

As a result M80 produces an output file, starting at a relative address of 0, but containing code that will execute at 0F000H. L80 will then create a suitable .COM file by invoking it in the following way:

```
L80 /p:100,monitor,monitor/n/e
```

The /p:100 is an instruction to L80 to load the file to address 100H, the default load address being normally 103H.

NOTE - If the ".phase" had been omitted, then to get all the program addresses adjusted suitably for 0F000H the required L80 invocation would be:

```
L80 /p:F000,monitor,monitor/n/e
```

which would result in an "out of memory" error message from L80, and no load of the file. However if the wanted address lay within the TPA (e.g. at 8000H), L80 would load the .REL file and create a .COM file, a LARGE .COM file, with the majority of the file (just under 32k, - addresses 100H-7FFFH) full of NULLs! These would have to be removed from the file before it could be presented to PROG.

A. APPENDIX - Disk Formats

The diskette normally supplied with the GM860 is in Gemini QDSS format. This may be read on all Gemini Galaxy systems, Gemini MultiBoard systems with GM825 disk sub-systems, all Quantum systems, Kenilworth 83G models, and Nascom systems fitted with the Gemini GM809/GM829 FDC board, GM825 disk sub-system and Gemini GM556 CP/M.

If you are unable to read Gemini QDSS format then you should return the diskette to the supplying Gemini dealer asking for it to be exchanged for:

a) a Gemini SDDS format disk (Nascom with GM805 disk sub-system)

b) a Gemini DDDS format disk (systems with GM815 disk sub-system)

c) a paper listing of the PROGINIT.MAC file

d) certain dealers MAY be able to offer alternative formats